All Codes:

TASK 1:

Mapper:

```
#!/usr/bin/env python


import sys


# Initialize a dictionary to store trip count and total distance per taxi
taxi_data = {}


for line in sys.stdin:
    # Split the line into fields
    fields = line.strip().split(',')


    # Check if the line has the expected number of fields
    if len(fields) >= 8:
        try:
            # Parse trip information
            taxi_id = fields[1]
            distance = float(fields[3])


            # Update the dictionary with trip count and total distance
            if taxi_id in taxi_data:
                trip_count, total_distance = taxi_data[taxi_id]
                trip_count += 1
                total_distance += distance
                taxi_data[taxi_id] = (trip_count, total_distance)
            else:
                taxi_data[taxi_id] = (1, distance)
        except ValueError:
            # Skip lines with invalid data
            continue

# Emit intermediate key-value pairs from the mapper
for taxi_id, (trip_count, total_distance) in taxi_data.items():
    print(f"{taxi_id}\t{trip_count},{total_distance}")
```

Reducer:

```
#!/usr/bin/env python

import sys

# Initialize variables to store trip count and total distance per taxi
current_taxi = None
trip_count = 0
total_distance = 0

for line in sys.stdin:
    # Split the line into key and value
    key, value = line.strip().split('\t')
    taxi_id, trip_info = key, value

    # Split trip_info into trip count and total distance
    count, distance = map(float, trip_info.split(','))

    if current_taxi == taxi_id:
        # Accumulate trip count and total distance for the current taxi
        trip_count += count
        total_distance += distance
    else:
        if current_taxi:
            # Calculate and emit the average distance per trip for the previous taxi
            average_distance = total_distance / trip_count
            print(f"{current_taxi}\t{trip_count},{average_distance}")

        # Reset variables for the new taxi
        current_taxi = taxi_id
        trip_count = count
        total_distance = distance

# Output the result for the last taxi
if current_taxi:
    average_distance = total_distance / trip_count
    print(f"{current_taxi}\t{trip_count},{average_distance}")
```

Shell Script:

```
#!/bin/bash

# Hadoop Streaming jar location
HADOOP_STREAMING_JAR="/usr/lib/hadoop/hadoop-streaming-2.10.1-amzn-1.1.jar"

# Input and output paths
INPUT_PATH="/input/Trips.txt"
OUTPUT_PATH="/output/task1"

# Remove previous output
hadoop fs -rm -r $OUTPUT_PATH

# Run Hadoop Streaming job
```

```
hadoop jar $HADOOP_STREAMING_JAR \
    -D stream.map.output.field.separator="\t" \
    -D stream.num.map.output.key.fields=2 \
    -D mapreduce.job.reduces=3 \
    -files task1_mapper.py,task1_reducer.py \
    -mapper "python3 task1_mapper.py" \
    -reducer "python3 task1_reducer.py" \
    -input $INPUT_PATH \
    -output $OUTPUT_PATH

# Display the output
 hadoop fs -cat $OUTPUT_PATH/part-*
```

TASK 2:

Mapper:

```
#!/usr/bin/env python

import sys
import random

# Read k and v from command line arguments
k = int(sys.argv[1])
v = int(sys.argv[2])

# Initialize medoids with random data points
medoids = []
data = []

for line in sys.stdin:
    fields = line.strip().split(',')
    trip_id = int(fields[0])
    pickup_x = float(fields[4])
    pickup_y = float(fields[5])

    if len(medoids) < k:
        medoids.append((trip_id, pickup_x, pickup_y))
    data.append((trip_id, pickup_x, pickup_y))

# Assignment step
for point in data:
    trip_id, pickup_x, pickup_y = point
    min_distance = float("inf")
    closest_medoid = None

    for medoid in medoids:
        medoid_id, medoid_x, medoid_y = medoid
        distance = (pickup_x - medoid_x) ** 2 + (pickup_y - medoid_y) ** 2

        if distance < min_distance:
            min_distance = distance
            closest_medoid = medoid_id

    print(f"{closest_medoid}\t{trip_id},{pickup_x},{pickup_y}")

# Output data for reducer
for point in data:
    trip_id, pickup_x, pickup_y = point
    print(f"{trip_id}\t{pickup_x},{pickup_y}")
```

Reducer:

```
#!/usr/bin/env python

import sys

# Identity reducer
for line in sys.stdin:
    print(line.strip())
```

Shell Script:

```
#!/bin/bash

# Set input and output paths
input_path="/input/Trips.txt"
output_path="/output/task2"

# Hadoop Streaming jar location
HADOOP_STREAMING_JAR="/usr/lib/hadoop/hadoop-streaming-2.10.1-amzn-1.1.jar"

# Remove previous output (if any)
hadoop fs -rm -r $output_path

# Run the MapReduce job
hadoop jar $HADOOP_STREAMING_JAR \
    -D mapreduce.job.reduces=3 \
    -files task2_mapper.py,task2_reducer.py \
    -mapper "python3 task2_mapper.py $1 $2" \
    -reducer "python3 task2_reducer.py" \
    -input $input_path \
    -output $output_path
```

```
# Display the final results
hadoop fs -cat $output_path/part-*
```

TASK 3:

Mapper 1:

```
#!/usr/bin/env python
import sys

# Loop through input lines
for line in sys.stdin:
    line = line.strip()
    data = line.split(',')

    # Check if the data has 4 elements and the first element is a digit
    if len(data) == 4 and data[0].isdigit():
        taxi_id, company, model, year = data
        print(f"{taxi_id}\tT\t{company}")

# Loop through input lines
for line in sys.stdin:
    line = line.strip()
    data = line.split(',')

    # Check if the data has 8 elements and the first element is a digit
    if len(data) == 8 and data[0].isdigit():
        trip_id, taxi_id, fare, distance, pickup_x, pickup_y, dropoff_x, dropoff_y = data
        print(f"{taxi_id}\tR\t1")
```

Reducer 1:

```
#!/usr/bin/env python
import sys

current_taxi_id = None
current_company = None
trip_count = 0

# Loop through input lines
for line in sys.stdin:
    line = line.strip()
    taxi_id, data_type, value = line.split('\t')

    # Check if the current taxi_id is different from the previous one
    if current_taxi_id != taxi_id:
        if current_taxi_id:
            # Emit data in the format: taxi_id    current_company
            print(f"{current_taxi_id}\t{current_company}")

        current_taxi_id = taxi_id
        current_company = None
        trip_count = 0

    # Determine if it's a 'T' (Taxi) or 'R' (Record) data type
    if data_type == 'T':
        current_company = value
    elif data_type == 'R':
        trip_count += int(value)

# Print the last taxi's data
if current_taxi_id:
    print(f"{current_taxi_id}\t{current_company}")
```

Mapper 2:

```
#!/usr/bin/env python
import sys

# Loop through input lines
for line in sys.stdin:
    line = line.strip()

    # Split the input line by the tab character ('\t')
    taxi_id, company = line.split('\t')

    # Emit data in the format: company    1
    print(f"{company}\t1")
```

Reducer 2:

```
#!/usr/bin/env python
import sys

current_company = None
trip_count = 0

# Loop through input lines
for line in sys.stdin:
    line = line.strip()
    company, count = line.split('\t')

    # Check if the current company is different from the previous one
    if current_company != company:
        if current_company:
            # Emit data in the format: current_company    trip_count
            print(f"{current_company}\t{trip_count}")

        current_company = company
        trip_count = 0

    # Accumulate trip counts
    trip_count += int(count)

# Print the last company's trip count
if current_company:
    print(f"{current_company}\t{trip_count}")
```

Shell Script:

```
#!/bin/bash

# Hadoop Streaming jar location
HADOOP_STREAMING_JAR="/usr/lib/hadoop/hadoop-streaming-2.10.1-amzn-1.1.jar"

# Define input and output directories
INPUT_PATH="/input"
OUTPUT_PATH="/output/task3"

# Remove previous output (if any)
hadoop fs -rm -r $OUTPUT_PATH
hadoop fs -rm -r /intermediate_output

# Run the first MapReduce job for the join operation
```

```
hadoop jar $HADOOP_STREAMING_JAR \
   -D stream.map.output.field.separator="\t" \
   -D stream.num.map.output.key.fields=2 \
   -D mapreduce.job.reduces=3 \
   -files task3_mapper1.py,task3_reducer1.py \
   -mapper "python3 task3_mapper1.py" \
   -reducer "python3 task3_reducer1.py" \
   -input $INPUT_PATH/Trips.txt $INPUT_PATH/Taxis.txt \
   -output /intermediate_output

# Run the second MapReduce job for counting
hadoop jar $HADOOP_STREAMING_JAR \
   -D stream.map.output.field.separator="\t" \
   -D stream.num.map.output.key.fields=2 \
   -D mapreduce.job.reduces=3 \
   -files task3_mapper2.py,task3_reducer2.py \
   -mapper "python3 task3_mapper2.py" \
   -reducer "python3 task3_reducer2.py" \
   -input /intermediate_output/part-* \
   -output $OUTPUT_PATH

# Display the final results
hadoop fs -cat $OUTPUT_PATH/part-*
```